PEMROGRAMAN JARINGAN

“Eksplorasi Mandiri 1”

DOSEN PENGAMPU:

Randi Proska Sandra, M.Sc.

OLEH:

Muhammad Alfarobi

23343011

PROGRAM STUDI INFORMATIKA

DEPARTEMEN TEKNIK ELEKTRONIKA

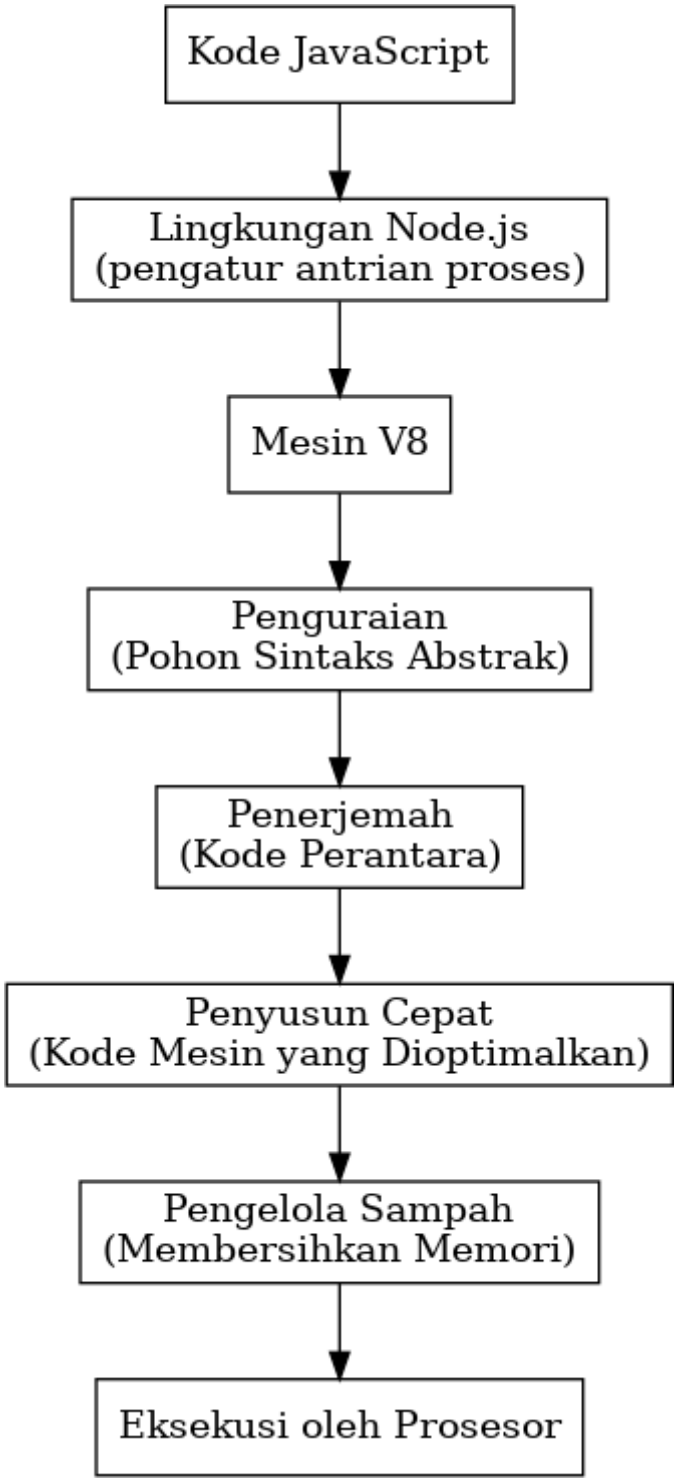
FAKULTAS TEKNIK

UNIVERSITAS NEGERI PADANG

2025

[V8 JavaScript Engine pada Node.js]

1. Diagram



Gambar 1. Diagram alur kerja V8 JavaScript Engine pada Node.js

2. Penjelasan diagram

Diagram tersebut menggambarkan tahapan eksekusi kode JavaScript di dalam lingkungan Node.js. Saat aplikasi JavaScript dijalankan, baris kode diteruskan ke lingkungan Node.js yang berfungsi sebagai manajer antrean proses (event loop). Setelah

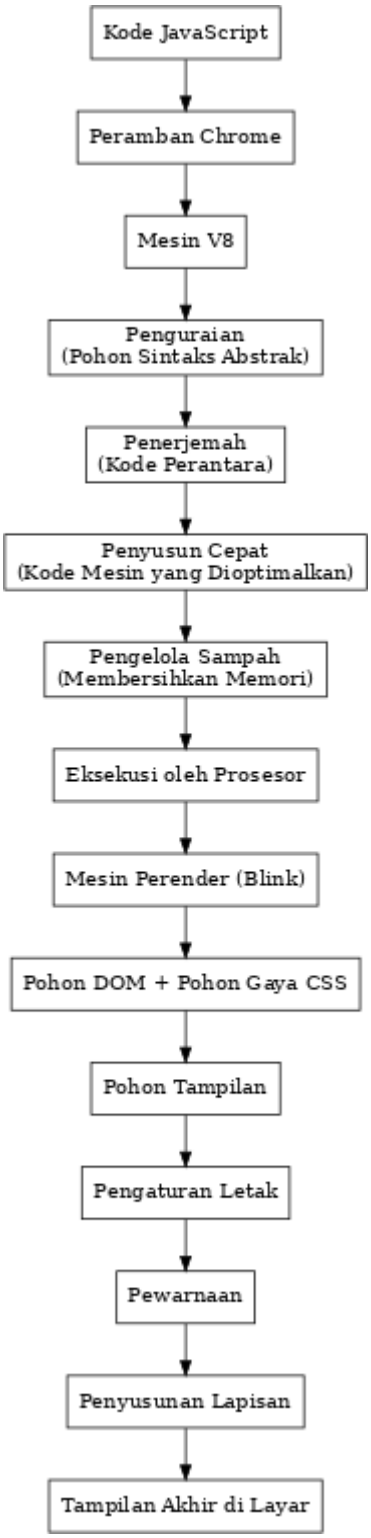
itu, kode dialihkan ke mesin V8, inti dari eksekusi JavaScript di Node.js. V8 pertama-tama melakukan parsing terhadap kode sumber, menghasilkan struktur data berupa Abstract Syntax Tree (AST) yaitu sebuah representasi pohon hierarkis dari program, sedangkan scanner dan parser menjalankan tugas tokenization dan sintaksis (token parsing) untuk membentuk AST.

Setelah AST terbentuk, V8 menerjemahkan struktur sintaks tersebut menjadi bytecode yang dijalankan oleh Ignition, yaitu interpreter ringan milik V8. Bytecode ini merupakan representasi menengah (intermediate representation) yang lebih efisien dibandingkan interpretasi langsung atas concrete AST yang memungkinkan startup cepat. Dari situ, apabila suatu bagian kode (hot path) dieksekusi berulang kali, V8 memanfaatkan data profil runtime untuk mengoptimasi kode: TurboFan (JIT compiler) mengubah bytecode tersebut menjadi machine code yang dioptimalkan, meningkatkan performa eksekusi secara dramatis.

Selain penerjemahan dan optimasi kode, diagram ini menunjukkan bahwa V8 juga mengelola aspek penting lain seperti pengelolaan sampah (garbage collection) yang membersihkan memori tidak terpakai, dan akhirnya meneruskan eksekusi ke prosesor. Garbage collector dijalankan secara otomatis oleh V8 untuk memelihara performa dan efisiensi memori. Setelah semuanya siap, instruksi machine code yang dihasilkan dari TurboFan dieksekusi langsung oleh prosesor, menutup seluruh rangkaian pipeline eksekusi JavaScript di Node.js.

[V8 JavaScript Engine pada browse]

1. Diagram



Gambar 2. Diagram alur kerja V8 JavaScript Engine pada browse

2. Penjelasan diagram

Diagram ini memperlihatkan bagaimana kode JavaScript dieksekusi dalam konteks peramban Chrome yang menggunakan mesin V8 sebagai inti pemrosesan JavaScript.

Pertama, kode JavaScript masuk melalui Chrome, lalu diteruskan ke V8 untuk dilakukan parsing, menghasilkan Abstract Syntax Tree (AST). AST dibangun melalui proses tokenisasi dan analisis sintaksis, yang memungkinkan struktur program dipahami secara hierarkis oleh mesin. Setelah AST terbentuk, V8 menggunakan interpreter untuk menghasilkan bytecode, sebuah representasi menengah dari program. Tahap ini memungkinkan startup aplikasi yang lebih cepat dibandingkan interpretasi langsung dari kode sumber.

Selanjutnya, bytecode dieksekusi oleh Ignition dan bagian kode yang sering dijalankan (hot path) dioptimalkan oleh TurboFan, just-in-time (JIT) compiler milik V8, sehingga diubah menjadi machine code yang efisien. Mesin juga mengaktifkan garbage collector untuk membebaskan memori dari objek yang sudah tidak terpakai. Setelah kode dioptimalkan menjadi kode mesin, instruksi ini dieksekusi oleh prosesor. Namun, karena ini berjalan dalam browser, hasil eksekusi tidak berhenti di CPU saja, melainkan diteruskan ke Blink, mesin perender Chrome, untuk memproses instruksi visual agar dapat ditampilkan ke layar.

Mesin Blink kemudian mengintegrasikan hasil eksekusi dengan Document Object Model (DOM) dan pohon gaya CSS. Dari gabungan tersebut terbentuk render tree atau pohon tampilan, yang akan melalui beberapa tahap penting: pengaturan tata letak (layout), pewarnaan (painting), hingga penyusunan lapisan (layer compositing). Proses ini memastikan setiap elemen halaman web ditempatkan pada posisi yang tepat dengan gaya visual yang benar. Akhirnya, hasil akhir dari semua proses tersebut adalah tampilan halaman web yang utuh pada layar pengguna, yang menjadi interaksi nyata antara kode JavaScript, mesin peramban, dan perangkat keras.

[Daftar Pustaka]

Indrajit, V. (2024, January 21). *Browser Architecture and In-depth understanding of the Javascript V8 engine*. Medium. Tersedia secara bebas.

Jindal, R. (2025, March 8). *Understanding Just-In-Time (JIT) Compilation in V8: A Deep Dive*. Medium.

Yasser, M. (2023). *The V8 Engine Series I: Architecture*. Medium.

Wang, E., Zurowski, S., Duffy, O., Thomas, T., & Baggili, I. (2022). *Juicing V8: A primary account for the memory forensics of the V8 JavaScript engine*. *Forensic Science International: Digital Investigation*, 42, 301400.